UNIT17: SKINNED MESH – FBX SDK

【学習要項】
☐FBX SDK
☐Scene
☐Node

【演習手順】
1．FBX SDK のセットアップ
①下記リンクから FBX SDK 2020.2 VS2019 と FBX SDK 2020.2 VS2019 PBDS をダウンロードする

```
https://www.autodesk.com/developer-network/platform-technologies/fbx-sdk-2020-2
```

②ダウンロードしたファイルを実行する

```
fbx20202_fbxsdk_vs2019_win.exe
fbx20202_fbxsdk_vs2019_pdbs.exe
```

③インストール先のディレクトリはデフォルトの状態のままにする

```
C:¥Program Files¥Autodesk¥FBX¥FBX SDK¥2020.2
```

④3dgp プロパティページの「C/C++」→「全般」→「追加のインクルード」に以下のパスを追加する
※構成「すべての構成」、プラットフォーム「すべてのプラットフォーム」
`C:¥Program Files¥Autodesk¥FBX¥FBX SDK¥2020.2¥include`

⑤3dgp プロパティページの「リンカー」→「全般」→「追加のライブラリ」に以下のパスを追加する
※構成「Debug」、プラットフォーム「x64」
`C:¥Program Files¥Autodesk¥FBX¥FBX SDK¥2020.2¥lib¥vs2019¥x64¥debug`

※構成「Release」、プラットフォーム「x64」
`C:¥Program Files¥Autodesk¥FBX¥FBX SDK¥2020.2¥lib¥vs2019¥x64¥release`

⑥3dgp プロパティページの「リンカー」→「入力」→「追加の依存ファイル」に以下のファイルを先頭に追加する
※構成「すべての構成」、プラットフォーム「すべてのプラットフォーム」
`zlib-md.lib;libxml2-md.lib;libfbxsdk-md.lib;`

2．skinned_mesh クラスの実装
①プロジェクトに skinned_mesh.h と skinned_mesh.cpp を追加する
②skinned_mesh.h に skinned_mesh クラスと scene 構造体を定義する

```
 1:  #pragma once
 2:  #include <d3d11.h>
 3:  #include <wrl.h>
 4:  #include <directxmath.h>
 5:  #include <vector>
 6:  #include <string>
 7:  #include <fbxsdk.h>
 8:  struct scene
 9:  {
10:      struct node
11:      {
12:          uint64_t unique_id{ 0 };
13:          std::string name;
14:          FbxNodeAttribute::EType attribute{ FbxNodeAttribute::EType::eUnknown };
15:          int64_t parent_index{ -1 };
16:      };
17:      std::vector<node> nodes;
18:      int64_t indexof(uint64_t unique_id) const
19:      {
20:          int64_t index{ 0 };
21:          for (const node& node : nodes)
22:          {
23:              if (node.unique_id == unique_id)
24:              {
25:                  return index;
26:              }
```

```
27:             ++index;
28:         }
29:         return -1;
30:     }
31: };
32: class skinned_mesh
33: {
34: public:
35:     struct vertex
36:     {
37:         DirectX::XMFLOAT3 position;
38:         DirectX::XMFLOAT3 normal;
39:         DirectX::XMFLOAT2 texcoord;
40:     };
41:     struct constants
42:     {
43:         DirectX::XMFLOAT4X4 world;
44:         DirectX::XMFLOAT4 material_color;
45:     };
46: private:
47:     Microsoft::WRL::ComPtr<ID3D11VertexShader> vertex_shader;
48:     Microsoft::WRL::ComPtr<ID3D11PixelShader> pixel_shader;
49:     Microsoft::WRL::ComPtr<ID3D11InputLayout> input_layout;
50:     Microsoft::WRL::ComPtr<ID3D11Buffer> constant_buffer;
51: public:
52:     skinned_mesh(ID3D11Device* device, const char* fbx_filename, bool triangulate = false);
53:     virtual ~skinned_mesh() = default;
54: protected:
55:     scene scene_view;
56: };
```

③skinned_mesh.cpp に skinned_mesh コンストラクタを実装する

※必要なヘッダファイルをインクルードすること

※動作確認後**#if-#endif** ディレクティブのコードは無効にすること**(41:-54:**行目**)**

```
 1: #include "misc.h"
 2: #include "skinned_mesh.h"
 3: #include <sstream>
 4: #include <functional>
 5: using namespace DirectX;
 6: skinned_mesh::skinned_mesh(ID3D11Device* device, const char* fbx_filename, bool triangulate)
 7: {
 8:     FbxManager* fbx_manager{ FbxManager::Create() };
 9:     FbxScene* fbx_scene{ FbxScene::Create(fbx_manager, "") };
10:
11:     FbxImporter* fbx_importer{ FbxImporter::Create(fbx_manager, "") };
12:     bool import_status{ false };
13:     import_status = fbx_importer->Initialize(fbx_filename);
14:     _ASSERT_EXPR_A(import_status, fbx_importer->GetStatus().GetErrorString());
15:
16:     import_status = fbx_importer->Import(fbx_scene);
17:     _ASSERT_EXPR_A(import_status, fbx_importer->GetStatus().GetErrorString());
18:
19:     FbxGeometryConverter fbx_converter(fbx_manager);
20:     if (triangulate)
21:     {
22:         fbx_converter.Triangulate(fbx_scene, true/*replace*/, false/*legacy*/);
23:         fbx_converter.RemoveBadPolygonsFromMeshes(fbx_scene);
24:     }
25:
26:     std::function<void(FbxNode*)> traverse{ [&](FbxNode* fbx_node) {
27:         scene::node& node{ scene_view.nodes.emplace_back() };
28:         node.attribute = fbx_node->GetNodeAttribute() ?
29:             fbx_node->GetNodeAttribute()->GetAttributeType() : FbxNodeAttribute::EType::eUnknown;
30:         node.name = fbx_node->GetName();
31:         node.unique_id = fbx_node->GetUniqueID();
```

```
32:         node.parent_index = scene_view.indexof(fbx_node->GetParent() ?
33:             fbx_node->GetParent()->GetUniqueID() : 0);
34:         for (int child_index = 0; child_index < fbx_node->GetChildCount(); ++child_index)
35:         {
36:             traverse(fbx_node->GetChild(child_index));
37:         }
38:     } };
39:     traverse(fbx_scene->GetRootNode());
40:
41: #if 1
42:     for (const scene::node& node : scene_view.nodes)
43:     {
44:         FbxNode* fbx_node{ fbx_scene->FindNodeByName(node.name.c_str()) };
45:         // Display node data in the output window as debug
46:         std::string node_name = fbx_node->GetName();
47:         uint64_t uid = fbx_node->GetUniqueID();
48:         uint64_t parent_uid = fbx_node->GetParent() ? fbx_node->GetParent()->GetUniqueID() : 0;
49:          int32_t type = fbx_node->GetNodeAttribute() ? fbx_node->GetNodeAttribute()->GetAttributeType() :
0;
50:         std::stringstream debug_string;
51:         debug_string << node_name << ":" << uid << ":" << parent_uid << ":" << type << "¥n";
52:         OutputDebugStringA(debug_string.str().c_str());
53:     }
54: #endif
55:     fbx_manager->Destroy();
56: }
```

3．framework クラスのメンバ変数として sknned_mesh *型配列を要素数 8 で宣言する

```
std::unique_ptr<skinned_mesh> skinned_meshes[8];
```

4．framework クラスの initialize メンバ関数で sknned_mesh オブジェクトを生成する

```
skinned_meshes[0] = std::make_unique<skinned_mesh>(device.Get(), ".¥¥resources¥¥cube.000.fbx");
```

5．実行し、VisualStudio の出力ウインドウに cube.000.fbx のシーンのノード情報が表示されることを確認する

6．別の FBX ファイル(plantune.fbx 等)をロードしてシーンのノード情報を確認する

【評価項目】
□FBX SDK の導入と初期化
□シーンノード情報の確認